Pandromeda

Fractal Models of Natural Phenomena

Dr. F. K. “Doc Mojo” Musgrave

CEO/CTO

Pandromeda, Inc.

Fractals

What is a Fractal?

- **A complex object**
- **The complexity of which derives from *self-similarity***
- **Or the repetition of form over a (finite) range of scales**

"Bigger swirls have smaller swirls

that feed on their velocity,

and smaller swirls have smaller swirls

and so on, to viscosity"

Fractals

Fractal dimension

- Generalization of familiar integer-valued dimension
- Fractal dimension is real-valued, e.g., 3.3
- Large value after decimal point ⇒ rough surface
- Small value after decimal point ⇒ smoother surface
- Fractal dimension is not mathematically well-defined
- Can be used entirely subjectively

Fractals
Dilation symmetry

- The easiest way to think of fractals:
  - *Dilation symmetry*
  - Invariance under change of scale (zooming in and out)
- Symmetry may be *exact* or *statistical*
  - Exact self-similarity: Koch snowflake
  - Statistical self-similarity: terrains, clouds

Fractals

Deterministic vs. random fractals

- **Deterministic fractals**
  - **Koch snowflake**
  - **Mandelbrot set**
- **Random fractals**
  - **Iterated function systems**
  - **L-systems**
  - **Fractional Brownian motion (fBm)**

Fractals

Complexity in Nature

- Nature is complex
- Fractals capture some—but not all—of that complexity
- Examples of fractals:

  Trees

  Mountains

  Turbulence: clouds, fire, smoke, astronomical jets
- Counterexample:

  A battered old tennis shoe

Fractional Brownian Motion (fBm)

What is it?

- **Generalization of *Brownian motion*:**
  - **Integral of progress on a random walk**
- **fBm is characterized by its *power spectrum***
  - **Brownian motion has $1/f^2$ power spectrum**
  - **fBm has $1/f^\beta$ power spectrum, $1.0 \leq \beta \leq 3.0$**
- **Just think of β as controlling roughness of the terrain**
- **For math, see Voss & Saupe in "The Science of Fractal Images"**

Pandromeda

Fractional Brownian Motion (fBm)

The key variables

- ***Basis function*:**
  The shape that is repeated over a range of scales
- ***Spectral exponent*:**
  Determines fractal dimension, or roughness of terrain
- ***Lacunarity*:**
  The gap between successive scales
- ***Octaves*:**
  The number of scales of self similarity

Fractional Brownian Motion (fBm)

The basis function

- Should have range [-1.0 . . 1.0]
  - So that integral remains zero
  - Expected value remains zero
- Shape is very important
  - Shape clearly shows through in fractal sum
  - (At lacunarity of 2.0)
- Can be literally anything!
  - Sparse convolution (*wavelets*) gives maximum flexibility
  - But is very expensive
  - (See Peachey in "Textures and Modelling: A Procedural Approach")

Fractional Brownian Motion (fBm)

The basis function

- Sine wave in Fourier synthesis
  - Mathematically pure: each frequency is defined exactly
  - Sine is periodic, so all finite sums of it are also periodic
- Triangle wave in polygon subdivision
  - Piecewise linear interpolation
  - Creases and sharp peaks
- Perlin noise
  - Piecewise cubic interpolation
  - Nice, aperidoic sine-wave substitute
- Others
  - Voronoi (see Worley, SIGGRAPH 96)
  - See list of basis functions in MojoWorld in CAL

Fractional Brownian Motion (fBm)

The basis function

- **Batch algorithms**
  - **Fourier synthesis**
  - **Polygon subdivision**

- **Point-evaulated**
  - **Perlin noise, Voronoi, sparse convolution**
  - **These are the so-called “procedural” methods**

- **Infinite support**
  - **Sine waves**
  - **Procedural noises**

- **Finite support**
  - **Polygon subdivision**
  - **Wavelets**

Fractional Brownian Motion (fBm)

The spectral exponent

- Determines the fractal dimension
- Or the roughness of our terrain
- Can be used correctly or incorrectly
- But you get a fractal nonetheless
  - See the course notes for the math
  - And the literature for unexpected complications
- But don't worry—use it qualitatively and ignore the math!

Pandromeda

Fractional Brownian Motion (fBm)

Lacunarity

- The gap between frequencies in spectral summation
- Virtually always set to 2.0 (hence "octaves")
- May want to use 2.0 ± ~0.1
  - To avoid artifacts in lattice-based noises
  - As with value & gradient Perlin noises
- Using values << 2.0
  - Slower: takes more octaves to get fine details
  - Gains little, visually
- Using values >> 2.0
  - Faster: takes less octaves to get fine details
  - But discrete frequencies can show through

Fractional Brownian Motion (fBm)

Octaves

- Number of octaves is number of scales of self similarity
- Octaves are only “octaves” when lacunarity is 2.0
- Octaves = detail
- Can be driven by Nyquist limit
  - To antialias by *clamping*
  - As in QAEB tracing (see “Textures and Modelling”)
  - Yields pixel-sized detail everywhere

Fractal Terrain Models

Different kinds

- ***Fourier***:
  - The most mathematically "pure"
  - Slow and periodic
- ***Polygon subdivision***:
  - Easiest to implement, but sports the worst artifacts
  - Triangle, square, nested, unnested, semi-nested
  - (see Miller, SIGGRAPH 86, and "The Science of Fractal Images")
- ***Point-evaluated / procedural***:
  - Can be the slowest, depending on basis function
  - Most flexible and, generally, best-looking

Fractal Terrain Models

Point-evaluated or procedural

- **Perlin noise fBm**

  Generalization of Perlin's "chaos" function

  (See Perlin, *An Image Synthesizer,* SIGGRAPH 85)

- **General procedural fBm**

  Same as above

  But using Voronoi noise, sparse convolution, etc.

- **Domain-distorted procedural fBm**

  Add a vector-valued function to the point, before evaluation

  Shmushes the resulting fractal around

Fractal Terrain Models

Point-evaluated or procedural

The basic algorithm:

1. **Start with lowest frequency (largest scale of basis)**
2. **Double the frequency**
3. **Scale amplitude down, according to spectral exponent**
4. **Add in new, scaled frequency**
5. **Goto 2.**

Code for Procedural fBm

```
fBm( Vector point,
     NoiseFunction basis(),
     real exponent, real lacunarity,
     integer octaves )
{
   real value = 0.0, amplitude = 1.0;

   for ( i=0; i<octaves; i++ ) {
   value += basis(point) * amplitude;
   point *= lacunarity;
   amplitude *= exponent;
   }

   return value;
}
```

Multifractals

Heterogeneous terrain models

- fBm is *stationary*: statistically homogeneous and isotropic

- Real terrain is far more complex
  - Mountains rise from plains
  - Peaks and valleys have different roughnesses, etc.

- We want to capture at least some of this
  - Devising heterogeneous fBm-based fractals
  - While preserving the elegance of fBm

Multifractals

Three multifractal terrain models

- **Stats-by-altitude**
  - Conjecture: valleys are smoother than peaks
  - Model: multiply each octave (after first) by current "altitude"
- **"Pure" multiplicative multifractal**
  - Inner loop is multiplicative, rather than additive as in fBm
  - Problem: converges to zero or diverges to infinity
- **Hybrid additive/multiplicative multifractal**
  - Conjecture: valleys should be smoother at all altitudes
  - Model: multiply octave *i* by value of octave *i* -1; sum this

Pandromeda

Code for Stats-by-Altitude Multifractal

```
StatsByAlt( Vector point, NoiseFunction basis(),
            real exponent, real lacunarity,
            integer octaves )
{
   real value, amplitude = 1.0;

   if ( octaves )    // do first octave
   value = basis( point );

   for ( i=1; i<octaves; i++ ) {
   value += value * basis(point) * amplitude;
   point *= lacunarity;
   amplitude *= exponent;
   }

   return value;
}
```

Pandromeda

Code for Multiplicative Multifractal

```
Multifractal( Vector point,
              NoiseFunction basis(),
              real exponent, real lacunarity,
              integer octaves )
{
   real value = 1.0, amplitude = 1.0;

   for ( i=0; i<octaves; i++ ) {
        value *= basis(point) * amplitude;
        point *= lacunarity;
        amplitude *= exponent;
   }

   return value;
}
```

Code for Hybrid Multifractal

```
HybridMF( Vector point, NoiseFunction basis(),
          real exponent, real lacunarity,
          integer octaves )
{
   real value, signal, weight, amplitude = 1.0;

   if ( octaves <= 0 ) return 0.0;

   weight = value = basis( point );     // first octave
   octaves -= 1.0;

   for ( i=1; i<octaves; i++ ) {
   signal = weight * basis(point) * amplitude;
   value += signal;
   weight = signal;
   point *= lacunarity;
   amplitude *= exponent;
   }

   return value;
}
```

Erosion

- Erosion is what shapes terrains
    - Bedrock is fractal; erosion works on this fractal substrate
    - Creates *context-sensitive* fractals: river networks
- Diffusive erosion
    - Dry creep, rain splash, animal activity, etc.
    - Equivalent to low-pass filter—can operate very efficiently
- Fluvial erosion: running water
    - Rivers and glaciers are principal geomorphic agents
    - Very important—but too hard to implement and slow to run!
    - (see Musgrave et al, SIGGRAPH 89, and geology literature)

Pandromeda

Conclusions

- Fractal models capture complexity, with simplicity
- *Amplification*: wealth of detail from simple model
- Height field terrain models don’t cut it
- fBm doesn’t cut it
- Multifractal models are a little better
- Dilation symmetry rocks!
- Alas, Nature is more complex than fractal geometry